AA214/AE296 – 1996 – Solutions for Problem Set 6

1. The trapezoidal method $u_{n+1} = u_n + \frac{1}{2}h(u'_{n+1} + u'_n)$ is used to solve the ODE $u' = \lambda u + a$ numerically.

   (a) What is the resulting OΔE ?

   ANSWER: Applying the representative ODE $u' = \lambda u + a$(Note this is the case with $\mu = 0$ from the lectures) to the trapezoidal method gives and collecting terms:

   $$(1 - \frac{1}{2}h\lambda)u_{n+1} = (1 + \frac{1}{2}h\lambda)u_n + ha$$

   (b) What is its exact numerical solution ?

   ANSWER: You can get this a number of ways. One can apply the scheme recursively and find the general relation or use the $P(E), Q(E)$ analysis from the notes. In any case the exact solution is

   $$u_n = \left(\frac{1 + \frac{1}{2}h\lambda}{1 - \frac{1}{2}h\lambda}\right)^n - \frac{a}{\lambda}$$

   (c) How does the exact steady state solution of the OΔE compare with the exact steady state solution of the ODE (Hint:The exact SS solution is $u(t \to \infty) = -\frac{a}{\lambda}$)?

   ANSWER: In this case the steady state OΔE solution is the same as the ODE solution.

2. Consider the ODE

   $$\mathbf{u}' = \frac{d\mathbf{u}}{dt} = [A]\mathbf{u} + \mathbf{f}$$

   with

   $$[A] = \begin{bmatrix} -10 & -0.1 & -0.1 \\ 1 & -1 & 1 \\ 10 & 1 & -1 \end{bmatrix} \quad , \quad \mathbf{f} = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}$$

   (a) Find the eigenvalues of $[A]$ using Matlab. What is the long time Steady State (SS) solution $\mathbf{u}$? How would the ODE solution behave in time? (Hint: Remember the $e^{\lambda t}$ form of ODE solutions.)

   ANSWER: Matlab gives the eigenvalues as $\lambda_1 = -9.8888$, $\lambda_2 = -0.1112$, and $\lambda_3 = -2.0000$, which means the long terms transients go to zero as $t \to \infty$. The steady-state solution is

   $$-[A]^{-1}\mathbf{f} = \begin{bmatrix} 0 \\ -5 \\ -5 \end{bmatrix}$$

   (b) Write a Matlab code to integrate from the initial condition $\mathbf{u}(0) = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ from time $t = 0$ for the three time advance schemes ($h = \Delta t$)

   i. $u_{n+1} = u_n + h(u')_n$ the Euler Explicit Scheme

   ii. $u_{n+1} = u_n + h(u')_{n+1}$ the Euler Implicit Scheme

   iii. $u_{n+\frac{1}{2}} = u_n + h(u')_n \quad ; \quad u_{n+1} = u_n + \frac{1}{2}h\left((u')_{n+\frac{1}{2}} + (u')_n\right)$ the Predictor-Corrector Scheme

   In all three cases use $h = 0.1$ for 1000 time steps, $h = 0.2$ for 500 time steps, $h = 0.4$ for 250 time steps and $h = 1.0$ for 100 time steps. Compare the computed SS solution with the exact SS solution.

   ANSWER: You should find that all the methods are stable and converge well to the steady-state solution for $h = 0.1, 0.2$, and that only the implicit scheme is stable and convergent for $h = 0.4, 1.0$. I've provided my Matlab code at the end of the answer sheet for comparison.

(c) Could you have predicted the behavior of the previous problem? In class we developed the $\sigma - \lambda$ relations for these methods.

i. For the Euler Explicit Scheme: $\sigma = (1 + h\lambda)$.

ii. For the Euler Implicit Scheme: $\sigma = 1/(1 - h\lambda)$.

iii. For the Predictor-Corrector Scheme: $\sigma = (1 + h\lambda + \frac{1}{2}(h\lambda)^2)$.

The stability condition is $|\sigma| \leq 1.0$. For the Euler Explicit scheme what is the predicted stability limit on h and is it confirmed by your Matlab code? (Hint: Try running just below and above the limit, also use the eigenvalues from 2(a) in the stability check).

ANSWER: The stability condition is that

$$|\sigma| = |1 + h\lambda| \leq 1.0$$

which leads to the inequality equation

$$-1 \leq 1 + h\lambda \leq 1$$

We have the trivial solution $h\lambda \leq 0$ which is automatically satisfied since $h \geq 0$ and all the $\lambda \leq 0$. The nontrivial solution is $-2 \leq h\lambda$ which if we check all three λ is satisfied is $h \leq \frac{2}{9.8888} = 0.202265$, you can check this by running $h = 0.202$ and $h = 0.203$ for 1000 time steps.

3. For the "backward differentiation" scheme given by

$$u_{n+1} = \frac{1}{3}\left[4u_n - u_{n-1} + 2hu'_{n+1}\right]$$

(a) The $\lambda - \sigma$ relation may be derived in the following manner. Applying the time-marching scheme to a representative equation of the form

$$\frac{du}{dt} = \lambda u + ae^{\mu t}$$

results in the following equation

$$u_{n+1} = \frac{1}{3}\left[4u_n - u_{n-1} + 2h\left(\lambda u_{n+1} + ae^{\mu h\,(n+1)}\right)\right]$$

and simple algebraic manipulation of the equation yields

$$(3 - 2\lambda h)\, u_{n+1} - 4u_n + u_{n-1} = 2hae^{\mu h\,(n+1)}$$

and on introducing the difference operator, E, the equation may be written as

$$\left[(3 - 2\lambda h)\, E - 4 + E^{-1}\right] u_n = [2hE]\, ae^{\mu hn}$$

The term in square brackets on the *LHS* is referred to as the characteristic polynomial, $P(E)$, and the term in square brackets on the *RHS* is referred to as the particular polynomial, $Q(E)$.

(b) The $\lambda - \sigma$ relation is given by

$$(3 - 2\lambda h)\,\sigma^2 - 4\sigma + 1 = 0$$

Solving for the roots of the characteristic polynomial gives

$$\sigma = \frac{2 \pm \sqrt{1 + 2\lambda h}}{3 - 2\lambda h}$$

(c) Using the power series expansion identity [1]

$$\sqrt{1+x} = 1 + \sum_{k=1}^{\infty} (-1)^{k-1} \, \frac{(|2k-3|)!!}{(2k)!!} \, x^k$$

the square root in the numerator of the solution for σ can be expanded in powers of λh

$$\sqrt{1+2\lambda h} = 1 + \sum_{k=1}^{\infty} (-1)^{k-1} \, \frac{(|2k-3|)!!}{(2k)!!} \, (2\lambda h)^k = 1 + \lambda h - \frac{1}{2}(\lambda h)^2 + \frac{1}{2}(\lambda h)^3 - \frac{5}{8}(\lambda h)^4 + \dots$$

Likewise, using the power series expansion identity (for $x^2 < 1$)

$$\frac{1}{1-x} = \sum_{k=0}^{\infty} x^k$$

the denominator of the solution for σ can be expanded in powers of λh

$$\frac{1}{3\left(1-\frac{2}{3}\lambda h\right)} = \frac{1}{3}\sum_{k=0}^{\infty}\left(\frac{2}{3}\lambda h\right)^k = \frac{1}{3} + \frac{2}{9}\lambda h + \frac{4}{27}(\lambda h)^2 + \frac{8}{81}(\lambda h)^3 + \frac{16}{243}(\lambda h)^4 + \dots$$

In this particular case the principal σ-root is had by taking the positive sign in the root equation. The Taylor series expansion for the principal σ-root is then

$$\begin{aligned}
\sigma_1 &= \left(3 + \lambda h - \frac{1}{2}(\lambda h)^2 + \frac{1}{2}(\lambda h)^3 - \dots\right)\left(\frac{1}{3} + \frac{2}{9}\lambda h + \frac{4}{27}(\lambda h)^2 + \frac{8}{81}(\lambda h)^3 + \dots\right) \\
&= 1 + \lambda h + \frac{1}{2}(\lambda h)^2 + \frac{1}{2}(\lambda h)^3 + \dots
\end{aligned}$$

Subtracting this from the Taylor series expansion of

$$e^{\lambda h} = 1 + \lambda h + \frac{1}{2}(\lambda h)^2 + \frac{1}{6}(\lambda h)^3 + \dots$$

and retaining only the first nonvanishing term, yields the transient error of the method

$$er_\lambda = e^{\lambda h} - \sigma_1 = -\frac{1}{3}(\lambda h)^3$$

(d) There is one spurious root which is had by taking the negative sign in the root equation. The first two nonvanishing terms in a Taylor series expansion of this spurious root are

$$\begin{aligned}
\sigma_2 &= \left(1 - \lambda h + \frac{1}{2}(\lambda h)^2 + \dots\right)\left(\frac{1}{3} + \frac{2}{9}\lambda h + \frac{4}{27}(\lambda h)^2 + \dots\right) \\
&= \frac{1}{3} - \frac{1}{9}\lambda h + \dots
\end{aligned}$$

4. For the time march method given by

$$u_{n+1} = u_{n-1} + \frac{2h}{3}(u'_{n+1} + u'_n + u'_{n-1})$$

[1] $(m)!!$ is the product of every other number, e.g., $(7)!! = 7 \cdot 5 \cdot 3 \cdot 1 = 105$, and $(6)!! = 6 \cdot 4 \cdot 2 = 48$.

(a) The $\lambda - \sigma$ relation may be derived in the following manner. Applying the time-marching scheme to a representative equation of the form ($\mu = 0$ in this case)

$$\frac{du}{dt} = \lambda u + a$$

results in the following OΔE

$$u_{n+1} = u_{n-1} + \frac{2h}{3}\left(\lambda u_{n+1} + \lambda u_n + \lambda u_{n-1} + 3a\right)$$

and on introducing the difference operator, E, the equation may be written as

$$\left[\left(1 - \frac{2h}{3}\lambda\right) E - \frac{2h}{3}\lambda + \left(1 + \frac{2h}{3}\lambda\right) E^{-1}\right] u_n = [2h]\, a$$

The term in square brackets on the *LHS* is referred to as the characteristic polynomial, $P(E)$, and the term in square brackets on the *RHS* is referred to as the particular polynomial, $Q(E)$.

(b) The $\lambda - \sigma$ relation is given by

$$\left(1 - \frac{2h}{3}\lambda\right)\sigma^2 - \frac{2h}{3}\sigma + \left(1 + \frac{2h}{3}\lambda\right) = 0$$

Solving for the roots of the characteristic polynomial gives

$$\sigma = \frac{\frac{h\lambda}{3} \pm \sqrt{1 - \frac{(\lambda h)^2}{3}}}{1 - \frac{2h\lambda}{3}}$$

Using the series expansions

$$\sqrt{1 - \frac{(h\lambda)^2}{3}} = 1 - \frac{(h\lambda)^2}{6} - \frac{(h\lambda)^4}{24} + \ldots$$

and

$$\frac{1}{1 - \frac{2h\lambda}{3}} = 1 + \frac{2h\lambda}{3} + \frac{4(h\lambda)^2}{9} + \frac{8(h\lambda)^3}{27} + \ldots$$

yields

$$\sigma_1 = 1 + h\lambda + \frac{1}{2}(h\lambda)^2 + \frac{1}{3}(h\lambda)^3 + \ldots$$

(c) Subtracting this from the Taylor series expansion of

$$e^{\lambda h} = 1 + \lambda h + \frac{1}{2}(\lambda h)^2 + \frac{1}{6}(\lambda h)^3 + \ldots$$

and retaining only the first nonvanishing term, yields the transient error of the method

$$er_\lambda = e^{\lambda h} - \sigma_1 = -\frac{1}{6}(\lambda h)^3$$

Although this wasn't request, one can note that the spurious root is

$$\sigma_2 \approx -1 - \frac{1}{9}\lambda h + \ldots$$

5. For the predictor-corrector combination

$$
\begin{aligned}
\tilde{u}_{n+1} &= u_n + hu'_n \\
u_{n+1} &= \alpha_1 u_n + \alpha_2 \tilde{u}_n + 1 + \beta h \tilde{u}'_{n+1}
\end{aligned}
$$

(a) The transient error can be minimized in the following way. Applying the time-marching scheme to the representative equation

$$\frac{du}{dt} = \lambda u + ae^{\mu t}$$

results in the following equation set

$$
\begin{aligned}
\tilde{u}_{n+1} &= u_n + h\left(\lambda u_n + ae^{\mu hn}\right) \\
u_{n+1} &= \alpha_1 u_n + \alpha_2 \tilde{u}_n + 1 + \beta h\left(\lambda \tilde{u}_{n+1} + ae^{\mu h\,(n+1)}\right)
\end{aligned}
$$

Introducing the difference operator, E, the equation set may be expressed in matrix form as

$$\begin{bmatrix} E & -(1+\lambda h) \\ -(\alpha_2 + \beta\lambda h)E & E - \alpha_1 \end{bmatrix} \begin{bmatrix} \tilde{u}_n \\ u_n \end{bmatrix} = \begin{bmatrix} h \\ h\beta E \end{bmatrix} ae^{\mu hn}$$

The characteristic polynomial equals the determinant of the matrix

$$P(E) = E\left(E - (\alpha_1 + \alpha_2) - (\alpha_2 + \beta)\lambda h - \beta(\lambda h)^2\right)$$

The nonzero root of this polynomial is

$$\sigma = (\alpha_1 + \alpha_2) + (\alpha_2 + \beta)\lambda h + \beta(\lambda h)^2$$

To minimize the transient error the following should hold

$$
\begin{aligned}
\alpha_1 + \alpha_2 &= 1 \\
\alpha_2 + \beta &= 1 \\
\beta &= \tfrac{1}{2}
\end{aligned}
$$

from which it readily follows that

$$\alpha_1 = \alpha_2 = \beta = \frac{1}{2}$$

so that the principal root is

$$\sigma_1 = 1 + \lambda h + \frac{1}{2}(\lambda h)^2$$

and the transient error is

$$er_\lambda = \frac{1}{6}(\lambda h)^3$$

(b) The particular polynomial, $Q(E)$, for the final family u_n (as opposed to the intermediate family $\tilde{u}_n$) is given by

$$Q(E) = \det \begin{bmatrix} E & h \\ -\frac{1}{2}(1+\lambda h)E & \frac{1}{2}hE \end{bmatrix} = \frac{1}{2}hE(E + 1 + \lambda h)$$

The exact numerical solution to $u' = \lambda u + ae^{\mu t}$ is then

$$
\begin{aligned}
u_n &= c_1 \sigma_1^n + ae^{\mu hn}\frac{Q(e^{\mu h})}{P(e^{\mu h})} \\
&= c_1\left(1 + \lambda h + \frac{1}{2}(\lambda h)^2\right)^n + ae^{\mu hn} \cdot \frac{\frac{1}{2}h(e^{\mu h} + 1 + \lambda h)}{e^{\mu h} - 1 - \lambda h - \frac{1}{2}(\lambda h)^2}
\end{aligned}
$$

6. The local phase error is defined as

$$er_\omega \equiv \omega h - \arctan \frac{\Im(\sigma)}{\Re(\sigma)}$$

and the power series expansion for the arctan function for $x^2 \leq 1$ is

$$\arctan x = \sum_{k=0}^{\infty} (-1)^k \frac{1}{2k+1} x^{2k+1}$$

(a) For the first order Runge-Kutta method the principal σ-root is

$$\sigma_1 = 1 + \lambda h$$

letting $\lambda = i\omega$, the local phase error becomes

$$\begin{aligned} er_\omega &= \omega h - \arctan(\omega h) \\ &= \omega h - \left[\omega h - \frac{1}{3}(\omega h)^3 + \frac{1}{5}(\omega h)^5 - \ldots \right] \\ &= \frac{1}{3}(\omega h)^3 \end{aligned}$$

(b) For the second order Runge-Kutta method the principal σ-root is

$$\sigma_1 = 1 + \lambda h + \frac{1}{2}(\lambda h)^2$$

letting $\lambda = i\omega$, the local phase error becomes

$$\begin{aligned} er_\omega &= \omega h - \arctan\left(\frac{\omega h}{1 - \frac{1}{2}(\omega h)^2} \right) \\ &= \omega h - \left[\frac{\omega h}{1 - \frac{1}{2}(\omega h)^2} - \frac{1}{3}\left(\frac{\omega h}{1 - \frac{1}{2}(\omega h)^2} \right)^3 + \frac{1}{5}\left(\frac{\omega h}{1 - \frac{1}{2}(\omega h)^2} \right)^5 - \ldots \right] \\ &= \omega h - \left[\sum_{k=0}^{\infty} \frac{(\omega h)^{2k+1}}{2^k} - \frac{1}{3}\left(\sum_{k=0}^{\infty} \frac{(\omega h)^{2k+1}}{2^k} \right)^3 + \frac{1}{5}\left(\sum_{k=0}^{\infty} \frac{(\omega h)^{2k+1}}{2^k} \right)^5 - \ldots \right] \\ &= \omega h - \left[\left(\omega h + \frac{(\omega h)^3}{2} + \frac{(\omega h)^5}{4} + \ldots \right) - \frac{1}{3}\left((\omega h)^3 + (\omega h)^5 + \ldots \right) + \frac{1}{5}\left((\omega h)^5 + \ldots \right) - \ldots \right] \\ &= \omega h - \left[\omega h + \frac{1}{6}(\omega h)^3 + \frac{7}{60}(\omega h)^5 - \ldots \right] \\ &= -\frac{1}{6}(\omega h)^3 \end{aligned}$$

(c) The first order method has a positive phase error, so it lags. Conversely, the second order method has a negative phase error, so it leads.

```
% main.m
%   Assignment 4  Problem 2, AE296  Spring 1995  Due March 13, 1995
A = [-10, -.1, -.1; 1, -1, 1; 10, 1, -1];
```

```
f = [-1 0 0]';
u0 = [1 1 1]';
uexact = A\(-f);

h = input('Enter time step = ');
N = input('Enter Number of time steps = ');

Lam = eig(A)

sig1 = max(abs(ones(3,1) + h*Lam));
sig2 = max(abs(ones(3,1)./(ones(3,1) - h*Lam)));
sig3 = max(abs(ones(3,1) + h*Lam + 0.5*h*h*Lam.*Lam));

sig1
sig2
sig3

uexp = u0;
uimp = u0;
upc  = u0;

ue2 = zeros(N,1);
ui2 = zeros(N,1);
up2 = zeros(N,1);
t = zeros(N,1);

for n = 1:N

  uexp = Euler_explicit(A,uexp,f,h);
  uimp = Euler_implicit(A,uimp,f,h);
  upc  = Predict_Correct(A,upc,f,h);

  ue2(n) = uexp(2);
  ui2(n) = uimp(2);
  up2(n) = upc(2);
  t(n) = n*h;
end

figure(1);
clf;
subplot(3,1,1);
plot(t,ue2,'r');
subplot(3,1,2);
plot(t,ui2,'r');
subplot(3,1,3);
plot(t,up2,'r');

diffe = uexp - uexact
diffi = uimp - uexact
diffp = upc - uexact
```

```
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Euler_explicit.m
function u = Euler_explicit(A,un,f,h)
up = uprime(A,un,f);
u = un + h*up;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Euler_implicit.m
function u = Euler_implicit(A,un,f,h)
up = uprime(A,un,f);
r = h*up;
u = un + (eye(3) - h*A)\r;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Predict_Correct.m
function u = Predict_Correct(A,un,f,h)
up = uprime(A,un,f);
ut = un + h*up;
upp = uprime(A,ut,f);
u = un + 0.5*h*(upp+up);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%uprime.m
function uprime = uprime(A,u,f)
uprime = A*u + f;
end
```